Semantic Web Spaces

Robert Tolksdorf, Lyndon Nixon, Franziska Liebsch,
Duc Minh Nguyen and Elena Paslaru Bontas

Freie Universität Berlin, Institut für Informatik
AG Netzbasierte Informationssysteme
http://nbi.inf.fu-berlin.de
Takustr. 9, D-14195 Berlin, Germany,

mailto:research@robert-tolksdorf.de, http://www.robert-tolksdorf.de
mailto:nixon@inf.fu-berlin.de, mailto:franziska@adestiny.de
mailto:nguyen@inf.fu-berlin.de, mailto:paslaru@inf.fu-berlin.de

July 16, 2004

Technical Report B-04-11

Abstract

In this paper we present an extension of XML-based tuplespaces for the Semantic Web. This proposal draws on the ideas of tuplespace computing as a paradigm closer to the basic principles of the Web. We describe the evolution of tuplespace computing, outline why tuplespaces are a suitable middleware for the Semantic Web and suggest how XMLSpaces can be extended to support Semantic Web technologies, like RDF(S) and OWL.

1 Introduction

The realization of the Semantic Web needs a set of specialized middleware as its infrastructure. Current middleware approaches are centralized in reasoners and are often not networked. There is a need for a higher abstraction which we propose as the "Semantic Web Space".

In the following we describe the evolution of tuple-space computing, explain how tuplespaces are a suitable model for the Semantic Web, suggest how a Semantic Web Space could be envisioned and describe its possible implementation as an extension of our XMLSpaces platform.

2 From Tuplespaces to Informationspaces for the Web

The coordination language Linda [Gelernter and Carriero, 1992] has its origins in parallel computing as a means to inject the capability of concurrent programming into sequential programming languages. For that, it defined a minimal language consisting of a handful of operations.

The idea was to define the high-level abstraction of a tuplespace in which data can be placed as tuples and retrieved by providing a template for which a match is sought. By making the retrieval operations blocking in the case of the absence of a match, Linda combines synchronization and communication in an extremely simple and abstract model. Linda is superior to traditional client-server models, since it uncouples interacting processes both in space and in time.

Despite the initially limited area of application in multiprocessor systems, Linda has attracted a variety of research over the past 20 years at various places, with the COORDINATION conference being a central event [Ciancarini and Hankin, 1996; Garlan and LeMetayer, 1997; Ciancarini and Wolf, 1999; Porto and Roman, 2000; Arbab and Talcott, 2002].

Aside from the theoretical foundation of Linda by a variety of formal models, attention has been on the application of Linda to different areas aside from parallel computing.

We illustrate some streams of that research with prior work of ours.

- *Using a space for service-oriented open systems*.

  Open distributed systems are computing systems that can be characterized by a heterogeneity of involved machine- and network architectures as well as of the data processed. They have to be able to integrate existing applications, to cope with the use of multiple programming-languages, and potentially high dynamics by joining and leaving components.

  The coordination language *Laura* [Tolksdorf, 1993; Tolksdorf, 1995; Tolksdorf, 1998] is designed to facilitate the use and offer of services in such a system. It is based on a shared collection of forms describing offers, requests, and results of services, called the service-space. Laura's operations permit the exchange of forms via the service-space, guided by a matching-rule based on a subtype-notion on service types. The uncoupled coordination paradigm inherited from Linda allows it to meet the requirements of open systems.

  With Laura, we apply Linda's coordination paradigm to open systems. Based on an analysis of the issue of names in open systems, we introduce a new approach

to typing of interfaces. The architecture we propose includes new extensions to a partial replication scheme that can cope with dynamically changing set of participating machines.

- *Using a space for coordination in the Web*.

  The original Web did not support multiuser, interactive applications. This shortcoming is being studied, and several approaches have been proposed. However, most existing approaches are oriented to centralized applications at servers, or local programs within clients. To overcome this deficit, we introduce PageSpace [Ciancarini *et al.*, 1996; Ciancarini *et al.*, 1998], that is both a reference architecture and a platform. We take a specific approach to coordinate agents in PageSpace, namely to use variants of the coordination language Linda to guide their cooperation. Coordination technology is integrated with the standard Web technology and the programming language Java.

  Several kinds of agents live in the PageSpace: User interface agents, personal home agents, the agents that implement applications, and the kernel agents of the platform. Within our architecture it is possible to support fault-tolerance and mobile agents as well.

- *Using spaces to coordinate workflows*.

  Web-based workflow management can greatly benefit from Internet technologies, as these offer required functionalities to support distributed and cross-organizational workflows. It is, however, vital for any successful Internet application to deliver its services via a standard protocol in a standard format.

  The *Workspaces* [Tolksdorf, 2002] architecture combines workflow management with standard Internet technology, namely the Extensible Markup Language XML and the Extensible Stylesheet Language XSL, with coordination technology. It is based on the notion of *steps* as the basic kinds of activity. Several kinds of steps describe activities or their coordination. A workflow definition is compiled into a set of steps that can be distributed individually. The implementation of Workspaces uses XSL processing for the generation and execution of steps.

  The Workspaces architecture and its implementation with standard Internet technologies is described. We report on the experiences made and draw conclusions on the adequacy of XSL as a platform independent and standardized component to build a Web-based workflow system.

- *XML-based coordination middleware with spaces*.

  There are several approaches to middleware with XML. One can distinguish coordination middleware with XML (the middleware uses XML to implement its services to applications), middleware for XML Agents embedded in documents (the agents are in XML, the middleware supports them), and self-contained XML Middleware (both the middleware and the application is XML-based) [Ciancarini *et al.*, 2003].

  XMLSpaces [Tolksdorf and Glaubitz, 2001] is a middleware with XML as an extension to the Linda model of coordination for Web-based applications. It allows XML documents to be stored in a coordination space from where they can be retrieved based on multiple matching relations amongst XML documents, including those given by XML query-languages. XMLSpaces is distributed and supports several distribution policies in an extensible manner.

XMLSpaces.NET [Tolksdorf *et al.*, 2004] implements the XMLSpaces concept as a middleware for XML documents on the .NET platform. It introduces an extended matching flexibility on nested tuples and richer data types for fields, including objects and XML documents. It is completely XML-based since data, tuples and tuplespaces are seen as trees represented as XML documents. XML-Spaces.NET is extensible in that it supports a hierarchy of matching relations on tuples and an open set of matching amongst data, documents and objects. In addition, the internal representation of the tuplespace is XML-based.

- *Selforganizing scalable implementations of spaces*.

  Traditional Linda implementations face enormous scalability problems. Natural forming multi-agent systems can grow to enormous sizes and perform seemingly complex tasks without the existence of any centralized control. Their success comes from the fact that agents are simple and the interaction with the environment and neighboring agents is local in nature. We describe how swarm intelligence can be used in the implementation of a Linda-based system called *SwarmLinda* [Menezes and Tolksdorf, 2004; Tolksdorf and Menezes, 2003]. In SwarmLinda we experiment with self-organization within the system, for example, by modelling templates seeking a matching tuple as ants seeking food and applying well-known scalable and decentralized algorithms to find it. We argue that SwarmLinda achieves many desired characteristics such as scalability, adaptiveness and even some level of fault-tolerance.

These approaches have applied the original Linda idea into several technological environments. The huge body of scientific literature on Linda contains several other experiments with extensions of Linda [Rossi *et al.*, 2001]. With relation to the Semantic Web, some directions seem of special interest:

- *Linda and logic programming*.

  Prolog-D-Linda [Sutcliffe and Pinakis, 1990; Sutcliffe and Pinakis, 1991] is an implementation of Prolog extended with Linda-style parallelism and supporting a distributed tuplespace. Tuples are expressed as Prolog clauses and tuplespaces as Prolog databases. Both Prolog facts and rules can exist in the space.

- *Linda and rules*.

  LGL (Law-Governed Linda) [Minsky and Leichter, 1995; Minsky *et al.*, 2000] applies the concept of law-governed architecture from the field of centralized message-passing systems to Linda in order to support the use of Linda as a coordination model for open systems.

  MARS [Cabri *et al.*, 2000] is a programmable coordination architecture for mobile agent applications. It defines Linda-like tuplespaces that react with specific actions to the accesses made by the mobile agents. The reactions can access the tuplespace, change its content and influence the semantics of the agents' accesses.

  TuCSon [Omicini and Zambonelli, 1999] is a coordination model for Internet applications based also on mobile agents. The model is based on the notion of "tuple centers", a tuple-based interaction space which enhances tuplespaces with programmable behavior. The approach differs from MARS in that it is used to deal with the heterogeneity, integrity and dynamics of information sources.

Linda has evolved from parallel programming into a very flexible concept that has successful been applied to various technical domains. There is a body of prior work that has already explored several directions that are relevant for the Semantic Web. In the following we propose a Linda-like system that could serve as a middleware for the Semantic Web.

3 A Semantic Web Space

3.1 Introduction of problem

The Semantic Web is a vision of a distributed network of machine-understandable knowledge. As it is built upon the existing Web infrastructure it inherits the Web architectural model, which has been formally described as REST (Representational State Transfer) [Fielding, 2000]. The fundamental principle of REST is that resources are stateless and published based on a global and persistent URI. The Semantic Web extends this world of URIs from addressable resources to any abstract concepts that must be represented within a computer system.

Tuplespaces have application to the Web in that they realize global places where information can be published and persistently stored. They have advantages over the standard client-server model in cases of parallel processing of published information from heterogeneous sources. Tuplespaces have been used to explore application to open distributed systems, multi-user and workflow co-ordination, XML middleware and self-organization. All of these areas are relevant to Web-based systems.

Semantic Web-based systems must also be able to meet requirements of reliability, scalability, self-organization, co-ordination with other systems and functioning over the open distributed system of the Web. Here tuplespaces can also be considered as a relevant paradigm. Semantic Web-based systems make use of access to knowledge stores distributed on the Web to acquire and infer knowledge and hence realize specific tasks. These knowledge stores must handle parallel access from multiple, heterogeneous systems, coordinating responses with other systems (e.g. that resolve ontological mismatches). Here we explore the idea of implementing a knowledge store for the Semantic Web as a tuplespace.

3.2 Requirements for a Semantic Web Space

Tuplespaces were designed primarily to function as local spaces for parallel processes within an application operating in a closed environment. To apply tuplespaces to the open global environment of the Web raises new requirements which have been mentioned in [Fensel, 2004; Johanson and Fox, 2004]:

- The naming of spaces, semantics and structure in describing the information content of the tuples. Otherwise tuples can not be distinguished from one another in terms of their contents when they have the same number of fields and field order.
- The nesting of tuples. Web data models such as XML permit the nesting of elements within a single document. Likewise Web-based information should be able to explicitly show where one unit is contained within another.
- A reference mechanism. The Web uses URIs as a global mechanism to uniquely address resources.

- A separation mechanism. Distributed applications which have independent naming schemes may use the same names for their tuplespaces, semantics or structure. On the Web, vocabularies can be kept separate – even when using the same terms – using the namespaces mechanism.

- Richer typing. Tuple values are typed according to the core data types. However this is not precise enough in a large scale environment with dynamically changing information. Richer typing can support validation and correct interaction with tuplespaces.

It has been proposed that using the Semantic Web standards – RDF and OWL – can be a solution to data and information heterogeneity just as the use of tuplespaces are a solution to protocol and process heterogeneity [Fensel, 2004]. This combination of Semantic Web and tuplespaces can be termed Semantic Web Spaces.

A Semantic Web Space contains RDF statements which are the core model of knowledge representation in the Semantic Web. These statements are represented in the tuplespace as tuples with the structure ⟨subject, predicate, object⟩. This proposal has been termed "triple-based computing" [Fensel, 2004]. By re-using the object of one statement as the subject of another, the tuplespace can represent a semantic graph structure of the contained knowledge.

The tuples shall be typed semantically. That is, the permitted values of the tuple fields shall be expressed as a class defined in an ontology. The ontology could use RDF Schema to define classes, properties, restrictions of the domain and range of properties and hierarchies of classes and properties. This expressiveness could be extended by the OWL ontology language which adds further ontological concepts.

Semantic typing will also include namespace support so that types from different vocabularies may be differentiated, also when they share the same name.

Each tuple in the Semantic Web Space should also be addressable through an unique identifier in order to group several semantically interrelated RDF statements. The unique identifier could be for example an automatically generated URI for an anonymous class, the URI of a reified statement or some internal identifier.

A Semantic Web Space will also require new matching procedures. Fundamentally, matching resources or identifying equivalent concepts with different URIs will require ontological-based reasoning. Different URIs may refer to the same concept, and different concepts may be identifiable as sub- or super-types of the given concept. Matching should be possible at different levels of precision and should have access to and make use of ontological knowledge to determine the relationships between classes and instances in the tuplespace.

3.3 Design of Semantic Web Space

Our initial design focuses on modelling the RDF semantic graph within a tuplespace. Just as the Semantic Web is conceived as a set of layers in which RDF is built upon by the subsequent layers of ontologies, logic, proof and trust, this initial design is only the beginning to further spaces for the Semantic Web:

- *RDFSpaces* will store RDF data and allow the storage and retrieval of triples by a set of extended matching relations that operate on the additional semantic information provided.

- *OWLSpaces* would permit the use of OWL concepts in the tuples as an extension of the RDF(S) concepts already used in the proposed Semantic Web Space. Due to the syntactical compatibility between OWL and RDF(S) every OWL construct can be represented as a uniquely identified set of RDF statements. The Semantic Web Space platform can interpret the knowledge according to the reasoning capabilities implemented in the application layer/matching functions. Different levels of expressiveness (OWL Lite, OWL DL etc) can be achieved by embedding corresponding reasoner modules in the application layer.

- *RuleSpaces* would make use of a Semantic Web rule language to explicitly represent laws that guide the interaction and are the basis to infer further information. These rules could be set as reactive to particular operations on the tuplespace, and could effect changes in the tuplespace additional to those carried out by agents.

- *ReasonSpaces* would include a logical reasoner for determining and publicizing proofs in the tuplespace. This could be based on e.g. PML to provide the proof to an agent that an operation result is correct.

- *TrustSpaces* would include agent policies as tuples and execute matchmaking agents to determine if two agents (source and target) can trust one another before permitting an operation.

The Semantic Web and Linda can be married in at least two ways: We can use the concept of Spaces to store Semantic Web information, the above ideas emphasize this approach. We could also use the Semantic Web to store metadata on Spaces like the semantic information that a specific tuplespaces stores travel information. This idea will be explored separately.

3.3.1 Representing RDF in a Space

To represent the RDF semantic graph, we permit within the tuplespace tuples with the ordered field types ⟨rdfs:Resource, rdf:Property, rdfs:Resource⟩. That is to say, we represent a RDF statement as a tuple and a RDF graph as a tuplespace. All RDF resources are represented in the tuplespace as URIs consisting of the namespace URI and a local name. QNames are not proposed to be used as the tuplespace would then need a facility to internally resolve namespaces to URIs.

In order to support rich typing, tuples in the Semantic Web Space are stored as an ordered list of typed field values. These values of these types are URIs identifying RDF classes constrained in a RDF Schema. In other words, each field value in the tuplespace is associated to a RDF type.

The semantic consistency of the tuplespace could be maintained by ensuring that all RDF resources are correctly validated according to their RDF type. As RDF schema information is representable as RDF, it follows that it is also representable within the tuplespace. Hence we propose that one functional component of the tuplespace platform will be to validate RDF statements being entered into the space by templates on the RDF Schema information available in the space for the field types in the statement.

3.3.2 Specifics of RDF Representation

There are three particular modelling constraints in RDF that need to be handled specifically in the tuplespace platform:

- *Blank nodes* are nodes in the RDF graph which do not have any form of URI identification. However associations tied to the same blank nodes must be both maintained in the tuplespace as well as represented by clients writing tuples or to clients reading tuples. We propose an extended RDF type called BlankNode, which can be instantiated in a tuple and given an unique URI value for representing that blank node in the local tuplespace, playing the role of "blank node identifier". This is still different than creating a newly URI-identified resource in that the URI of a BlankNode is not considered globally valid. An application receiving an URI as an instance of BlankNode should know that the URI is only to be considered as valid within the scope of the queried tuple space. Actually, we expect that applications reading from/writing to the Semantic Web Space will store information locally in RDF, and so the BlankNode type simply functions as a representable abstraction of blank nodes for the tuplespace.

- *Containers and collections* are special RDF objects which represent a set of resources. We propose that the members of rdfs:Container typed resources (rdf:Bag, rdf:Alt, rdf:Seq) are represented by a resizable array datatype, and that the members of a collection (rdf:List) are represented by a closed array datatype. In the tuplespace the container or collection can be referenced as a blank node or by an URI, and is related to its members through a statement with the rdfs:member property. Note that this approach represents the RDF graph for containers and collections while providing for a more efficient storage of statements by using built-in programming language datatypes. In a RDF serialization the membership property of a container is expanded into the individual properties rdf:_1, rdf:_2, rdf:_3 and so on with each member as object respectively. In the case of a collection, membership is expanded into rdf:first and rdf:rest propertied statements.

- *Reification* is the means by which a statement can be made about another statement in RDF. For this, RDF uses its own vocabulary (rdf:Statement, rdf:subject, rdf:predicate, rdf:object) to identify a statement with an URI so that it can be used as a subject or object of another statement. We propose the use of nested tuples to represent reification in the tuplespace. In this way a statement (as a tuple) can be the value of a field in another tuple. Nested tuples are considered as instances of the RDF type rdf:Statement with a BlankNode URI as identifier. This suffices for reference to a statement within the scope of the local tuplespace as the statement does not have a global URI for identification. Rather statements about the same statement are to be resolved by the tuplespace by associating the nested tuple to the same BlankNode reference. However we note that reference to a reified statement can be obtained by an agent by an appropriate query in the tuplespace e.g. if Z is a reified statement in the tuple $\langle X,Y,Z \rangle$ then reference can be obtained with the template $\langle X,Y,? \rangle$. So we propose that a tuple with a nested tuple written to the tuplespace considers the reified statement as a new BlankNode in the RDF graph *unless* the nested tuple is a tuple read out of the same tuple space and hence already identified in that tuplespace through a BlankNode instance.

3.3.3 Representing OWL in a Space

The OWL extension of the RDFSpaces is a straight forward task if we consider the syntactical compatibility of the two languages: OWL is expressed through RDF triples.

Every OWL construct can be serialized as a group of interrelated RDF statements and therefore can be represented in an RDFSpace. OWL restrictions are represented using blank nodes. owl:unionOf, owl:oneOf, owl:intersectionOf and owl:AllDifferent require a RDF collection. The previous section has already iterated how this could be represented in the tuplespace. Hence, representing OWL is considered a trivial extension of the RDFSpace.
The semantic differences between OWL and RDF Schema play a role only for the matching algorithms, which can use reasoners with specific capabilities.

3.3.4 Identification and Grouping

An issue arising from reification is the lack of an addressing scheme for the tuples themselves. While we believe the use of nested tuples and internal references can model the reification of statements in the RDF graph, an universal addressing scheme is a relevant requirement for the Semantic Web Space. [Johanson and Fox, 2004] note as a requirement for an open environment such as an interactive workspace a "standard routing field" to support a routing mechanism between agents. In their system they add a source field with information about the source of the tuple and a target field on templates with information about the querying receiver. We are seeking to apply the Web paradigm (which has been formalized as REST) through tuplespaces to the Semantic Web. Internal identifiers are assigned to RDF statements in order to conform to the REST formalization (in case of reified statements the identifiers are URIs) . We propose the extension of the triple-based approach to a quad-based approach (i.e. the addition of an optional fourth field in tuples in the Semantic Web Space). This additional field functions as an identifier for the tuple, either indicating the tuples source if the source is a knowledge provider or an intended target if the tuple's source is a knowledge requester. We consider the addition of this field an important design issue of the Semantic Web Space in order to support tuple distribution, routing, addressing and querying.

The Semantic Web Space must reflect the open and distributed nature of the Web, which requires a distributed approach to tuplespaces which can support stability and scalability at a very high magnitude. A decentralized architecture for a Semantic Web Space involves supporting the virtual integration of tuplespaces into a single view on the combined tuplespace. The distribution of tuples into this shared tuplespace could be organized according to thematic relations. In other words, we could partition our Semantic Web Space by forming "related" statements (i.e. referring to a similar set of concepts/individuals) into clusters that should be kept together.

3.3.5 Services in an Active Space

We also propose to support active services through the tuplespace (as an extension of Semantic Web, this is a realization of Semantic Web Services). To do this, the approach to services-based co-ordination of Laura may be applicable. Tuples represent service offers, service requests and service responses. In our case these tuples are also RDF statements which are representing respectively the service capabilities, the knowledge as input to the service and the knowledge as output from the service. We can utilize the fourth (identifier) field of the tuples for service identification and process monitoring. A client uses a query on the tuplespace to match its requirements with an available service capability and can write a service request to the tuplespace with that service as the intended target. The service which monitors the tuplespace for service requests

can read the tuple, and respond to it with a service response. The requesting client is monitoring the tuplespace for the response. Interactions between client and service can be followed by using an agreed upon URI scheme to identify a particular conversation (something like a session id appended to the services base URI). Note that this use of URIs for each stage of the interaction conforms to the proposals for Web Services based on the REST architecture.

3.3.6 Matching

Semantic Web Services, or any application built upon the use of the Semantic Web Space, are made possible as a result of appropriate and correct matching mechanisms. Matching through templates is the fundamental interaction paradigm of tuplespaces. New matching procedures are required to support the RDF-based paradigm of the Semantic Web Space which includes matching not only on values but also on types. As well as resource matching (equivalence of not only simple datatypes but also complex datatypes (arrays, lists) and of URIs), matching procedures must also take into account RDF Schema information and different levels of precision matching made available to allow requesters to choose the bounds in which a template may be determined as valid for a tuple.

For example, the template ⟨?,Y,?ex:Animal⟩ means match on tuples whose subject has the predicate Y and the object of type Animal. Given a tuple ⟨X,Y,Z⟩ where Z is of type Dog and the RDF Schema information that Dog is a subclass of Animal, we expect this tuple to match. However the matching will only succeed if the matcher is aware of the RDF Schema information, and the matching procedure selected allows matching on subclasses. The matching functionality for clients will need to support simple templates as well as queries in a suitably expressive RDF query language which could be mapped into tuplespace templates. Relationships between templates and nesting of templates need to be expressible (e.g. find tuples in which resource R is an object in one statement and the subject in another, or find tuples in which resource R is the object of a statement whose subject is the result of another template).

Querying and query responses also need to know how to handle blank nodes (local field values) since they do not have global URI identification. Reference to them can be acquired by a preliminary query in the tuplespace. Reified statements could be differentiated by two different levels of matching, one which considers nested tuples as tuples in their own right (and hence a possible response to a template) and another which only matches on the first level of tuples, ignoring nested tuples. OWL primitives and their semantical interpretations are handled by embedding available DL reasoners.

A relevant question on matching concerns *fairness* of matches. There are no guarantees whatsoever on how Linda finds the results for a retrieval operation. It has to be studied whether there are *fairness guarantees* on the usage of information and knowledge that are important. It has to be studied whether there is some notion of *safety* of a Semantic Web Space which ensure that a wrong statement posted to the space (⟨1+1, equals, 3⟩) does not infect the space in general.

4 Implementation of Semantic Web Space

4.1 XMLSpaces as an Implementation Basis

A Semantic Web Space could be built as an extension of our XMLSpaces platform, just as the Semantic Web (starting with RDF) is conceived as being built upon XML.

XMLSpaces is a distributed coordination platform that extends the Linda coordination language with the ability to carry XML documents in tuple fields. It introduces an extended matching flexibility on nested tuples and richer data types for fields, including objects and XML documents. It support an extensible set of matching relations as a hierarchy of matching relations on tuples and an open set of matching amongst data, documents and objects.

XMLSpaces has first been implemented in Java and then refined as XMLSpaces.NET in C# on the .NET platform. It is completely XML-based since data, tuples and tuplespaces are seen as trees represented as XML documents. The implemented distribution strategy is also extensible.

The following arguments make XMLSpaces a suitable implementation platform for Semantic Web Spaces [Tolksdorf and Glaubitz, 2001; Tolksdorf *et al.*, 2004]:

- The XMLSpaces platform has been itself built as an extension of the Linda coordination language to support XML documents as tuple fields and to extend matching routines with specifics for relations between XML documents. Hence there is already experience in extending a tuplespace to handle a new data model.
- The XMLSpaces platform architecture has been designed to be extensible. Hence extension is possible to new types of tuple field values and to new forms of matching.
- XMLSpaces supports open distribution of resources in order to support the coordination of wide area applications. It supports the integration of multiple servers into a single logical dataspace. Its distribution strategy can be configured at startup, and current implementations include full and partial replication.
- XMLSpaces views the tuplespace conceptually as a single XML tree with tuples forming subtrees and their fields forming the leaves. Hence there is nothing new in XMLSpaces with viewing the tuplespace in terms of a structured data model. The approach that was taken to support the conceptualization of the tuplespace as a XML tree could also be applied to conceiving the tuplespace as a RDF graph.
- Tuples in Linda are “primitive data” – there are no higher order values such as nesting of tuples. XMLSpaces introduced nested tuples to allow for a richer structuring of tuples in the XML tree.
- Typing of fields can also be built onto the XMLSpaces platform. While richer data typing has been abstracted to the XML document level, the current implementation also permits the use of objects from a programming language as a tuple value. Object serialisation and matching is object class specific, where the object class and its properties and relationships are defined within the programming code. A similar approach could be taken for the use of conceptual instances, where class definitions are resolved from an ontology and used for serialization and matching.

- Namespace and URI support can be added to the platform. Both are already aspects of the XML support required of the platform in allowing well-formed and valid XML to be the value of tuple fields. (URIs are datatypes in the XML Schema and XML Namespaces are supported in the DOM Level 2 object model)

- Matching procedures for a Semantic Web Space will need to support different levels of precision and be able to resolve heterogeneity and perform ontological reasoning. Experience with matching relations has already been had through the XMLSpaces implementation which required a variety of matching relations: matching on a shared schema, on a minimal subset of the schema, on equal contents, on equality of attributes in elements, or on a query expression. Some of these matching procedures needed the interpretation of external schema documents to determine a match. Likewise, in Semantic Web matching, different levels of match need to be offered and some will require interpretation of an external ontology document.

4.2 Necessary Steps to Implement Semantic Web Spaces with XMLSpaces

The Semantic Web Space can be built from the XMLSpaces middleware platform, as this already implements much required functionality of a distributed tuplespace. However, many implementation changes are also necessary which we note from the functional differences between the XMLSpaces platform and that of the proposed Semantic Web Space:

- Tuples in a Semantic Web Space are constrained to three or optionally four fields. The first three field types are constrained to being valid RDF types (rdfs:Resource as subject and object, rdf:Property as predicate) and the optional fourth to being an ID type. Field types are defined in RDF Schema and a reasoner should be able to access this schema information in order to be able to validate the tuples in the tuplespace.

- New matching procedures shall be implemented based on ontological reasoning about classes and properties (using RDF Schema and OWL information). They shall carry out the matching between the first three fields of the template and the first three fields of the tuples. The ID matching on the optional fourth field is implemented as a separate operation. A mapping between a RDF/OWL query language and tuplespace templates must be defined.

- The Semantic Web Space has a view on the tuplespace not as a XML DOM but as a RDF graph (e.g. in a Java implementation a switch from JAXP to Jena). The RDF implementation must be extended to include a type BlankNode and support the use of programming language datatypes for Containers and Collections. The Semantic Web Space also requires a different handling of nested tuples as that currently in XMLSpaces (the use of internal references, matching possible directly on a nested tuple).

- The reading and writing of tuples should be possible by the transformation to and from raw RDF (as XML serialization, N3 triples, etc) within the platform to and from sets of tuples, including handling the special cases of reification, collections/containers and blank nodes.

- The platform shall include the possibility of extensibility to a services paradigm, as well as to the inclusion of rules, proofs and trust. A services-based co-ordination will need support for persistent querying (a template remains valid over time for a given agent)

The vast majority of Semantic Web components is written in Java. There has to be some efficient bridge between the .NET world and the Java world. This might be a direct bridge to the use of traditional Web Services. XMLSpaces.NET currently gets a Web Services interface, so this implementation path might be easy to follow.

For an implementation, it will be very important to evaluate the performance of such a system when dealing with very large tuplespaces. After all, any Web-based technology will have to be able to cope with the astonishing size and scale of the Web if it is to be capable of a concrete deployment.

5 Outlook

This paper has introduced the Semantic Web Space, using the paradigm of tuplespaces to realize a necessary middleware for activity on the Semantic Web. We believe the Semantic Web Space offers a scalable co-ordination model based on the Linda parallel computing approach for supporting multiple agents participating in the provision and acquisition of knowledge.

Tuplespaces are considered a suitable model through their support for the web paradigm of an open, distributed system populated with resources identified by global and persistent URIs. There exists a plethora of extensions to Linda, including one for service-oriented systems, interactive workflow management and variants with richer expressibility like XMLSpaces.

A proposed means to realize the Semantic Web Space has been outlined and its possible implementation based on our existing implementation of a XML-based co-ordination middleware has been described. The basic steps to generate a conceptual model and prototype Semantic Web Spaces have been described.

References

[Arbab and Talcott, 2002] F. Arbab and C. Talcott, editors. *5th Int. Conf. on Coordination Languages and Models (COORDINATION)*, volume 2315 of *Lecture Notes in Computer Science*, York, UK, April 2002. Springer-Verlag, Berlin.

[Cabri *et al.*, 2000] G. Cabri, L. Leonardi, and F. Zambonelli. MARS: a Programmable Coordination Architecture for Mobile Agents. *IEEE Internet Computing*, 4(4):26–35, 2000.

[Ciancarini and Hankin, 1996] P. Ciancarini and C. Hankin, editors. *1st Int. Conf. on Coordination Languages and Models (COORDINATION)*, volume 1061 of *Lecture Notes in Computer Science*, Cesena, Italy, April 1996. Springer-Verlag, Berlin.

[Ciancarini and Wolf, 1999] P. Ciancarini and A. Wolf, editors. *3rd Int. Conf. on Coordination Languages and Models (COORDINATION)*, volume 1594 of *Lecture Notes in Computer Science*, Amsterdam, April 1999. Springer-Verlag, Berlin.

[Ciancarini *et al.*, 1996] Paolo Ciancarini, Andreas Knoche, Robert Tolksdorf, and Fabio Vitali. PageSpace: An Architecture to Coordinate Distributed Applications on the Web. *Computer Networks and ISDN Systems*, 28(7–11):941–952, 1996. Proceedings of the Fifth International World Wide Web Conference.

[Ciancarini *et al.*, 1998] Paolo Ciancarini, Robert Tolksdorf, Fabio Vitali, Davide Rossi, and Andreas Knoche. Coordinating Multiagent Applications on the WWW: A Reference Architecture. *IEEE Transactions on Software Engineering*, 24(5):362–375, May 1998. Special Issue: Mobility and Network Aware Computing.

[Ciancarini *et al.*, 2003] Paolo Ciancarini, Robert Tolksdorf, and Franco Zambonelli. Coordination Middleware for XML-centric Applications. *Knowledge Engineering Review*, 17(4), 2003.

[Fensel, 2004] D. Fensel. Triple-based computing. http://www.wsmo.org/2004/tp-computing/, June 2004.

[Fielding, 2000] R. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California Irvine, 2000.

[Garlan and LeMetayer, 1997] D. Garlan and D. LeMetayer, editors. *2nd Int. Conf. on Coordination Languages and Models (COORDINATION)*, volume 1282 of *Lecture Notes in Computer Science*, Berlin, Germany, September 1997. Springer-Verlag, Berlin.

[Gelernter and Carriero, 1992] David Gelernter and Nicholas Carriero. Coordination Languages and their Significance. *Communications of the ACM*, 35(2):97–107, 1992.

[Johanson and Fox, 2004] B. Johanson and A. Fox. Extending Tuplespaces for Coordination in Interactive Workspaces. *Journal of Systems and Software*, 69(3):243–266, 2004.

[Menezes and Tolksdorf, 2004] Ronaldo Menezes and Robert Tolksdorf. Adaptiveness in Linda-based Coordination Models. In *Proceedings of the First International Workshop on Engineering Self-Organising Applications (ESOA 2003)*, number LNCS 2977. Springer, 2004.

[Minsky and Leichter, 1995] Naftaly H. Minsky and Jerrold Leichter. Law-Governed Linda as a Coordination Model. In P. Ciancarini, O. Nierstrasz, and A. Yonezawa, editors, *Workshop on Languages and Models for Coordination, European Conference on Object Oriented Programming*, LNCS 924, 1995.

[Minsky *et al.*, 2000] N.H. Minsky, Y.M. Minsky, and V. Ungureanu. Making Tuple Spaces Safer for Heterogeneous Distributed Systems. In *Proceedings of the 2000 ACM Symposium on Applied Computing (SAC00)*, COMO, Italy, March 2000.

[Omicini and Zambonelli, 1999] A. Omicini and F. Zambonelli. Coordination for Internet Application Development. *Autonomous Agents and Multi-Agent Systems*, 2(3):251–269, 1999.

[Porto and Roman, 2000] A. Porto and GC. Roman, editors. *4th Int. Conf. on Coordination Languages and Models (COORDINATION)*, volume 1906 of *Lecture Notes in Computer Science*, Cyprus, September 2000. Springer-Verlag, Berlin.

[Rossi *et al.*, 2001] Davide Rossi, Giacomo Cabri, and Enrico Denti. Tuple-based Technologies for Coordination. In Andrea Omicini, Franco Zambonelli, Matthias Klusch, and Robert Tolksdorf, editors, *Coordination of Internet Agents: Models, Technologies, and Applications*, chapter 4, pages 83–109. Springer Verlag, 2001. ISBN 3540416137.

[Sutcliffe and Pinakis, 1990] G. Sutcliffe and J. Pinakis. Prolog-Linda – An Embedding of Linda in muProlog. In C.P. Tsang, editor, *Proceedings of the AI'90 – the 4th Australian Conference on Artificial Intelligence*, pages 331–340, 1990.

[Sutcliffe and Pinakis, 1991] Geoff Sutcliffe and James Pinakis. Prolog-D-Linda: An Embedding of Linda in SICStus Prolog. Technical Report 91/7, The University of Western Australia, Department of Computer Science, 1991.

[Tolksdorf and Glaubitz, 2001] Robert Tolksdorf and Dirk Glaubitz. Coordinating Web-based Systems with Documents in XMLSpaces. In *Proceedings of the Sixth IFCIS International Conference on Cooperative Information Systems (CoopIS 2001)*, number LNCS 2172, pages 356–370. Springer Verlag, 2001.

[Tolksdorf and Menezes, 2003] Robert Tolksdorf and Ronaldo Menezes. Using Swarm Intelligence in Linda systems. In Andrea Omicini, Paolo Petta, and Jeremy Pitt, editors, *Proceedings of the Fourth International Workshop Engineering Societies in the Agents World ESAW'03*, 2003.

[Tolksdorf *et al.*, 2004] Robert Tolksdorf, Franziska Liebsch, and Duc Minh Nguyen. XMLSpaces.NET: An Extensible Tuplespace as XML Middleware. In *Proceedings of the 2nd International Workshop on .NET Technologies, .NET Technologies'2004*, 2004.

[Tolksdorf, 1993] Robert Tolksdorf. Laura: A Coordination Language for Open Distributed Systems. In *Proceedings of the 13th IEEE International Conference on Distributed Computing Systems ICDCS 93*, pages 39–46, 1993.

[Tolksdorf, 1995] Robert Tolksdorf. *Coordination in Open Distributed Systems*. Number 362, Reihe 10 in VDI Fortschrittsberichte. VDI Verlag, 1995. ISBN 3-18-336210-4, PhD Thesis.

[Tolksdorf, 1998] Robert Tolksdorf. Laura—A service-based coordination language. *Science of Computer Programming*, 31(2–3):359–381, July 1998.

[Tolksdorf, 2002] Robert Tolksdorf. Workspaces: A Web-Based Workflow Management System. *IEEE Internet Computing*, 6(5):18–26, 2002.